1J－2

広域分散環境下における移動型プログラム構築環境の実装

古閑直樹*, 石田慶樹**, 牛島和夫†
*九州大学工学部情報工学科 **九州大学大型計算機センター
†九州大学大学院システム情報科学研究科

1 はじめに

近年、複数の計算機がネットワークを介して接続された分散環境下における移動型プログラムに関する研究が進んでいる[1, 2, 3]。移動型プログラムとは、ある計算機から別の計算機へ移動し、移動先の計算機で動作するプログラムのことで、モーバイルエージェントとも呼ばれている[1]。

移動型プログラムは各所で研究されているとは言え[2][3]、これらが十分な機能を備えているとは言い難い。既存の移動型プログラムは、自身の構成部品の一部が別の計算機に送られ動作し結果を返すだけであり、プログラム全体は移動しない。また、移動先の計算機で移動前の処理を再開できるように各種状態を保存・復元する機構を持つものは少ない。状態の保存・復元機構の実装は、移動型プログラムが動作する各計算機のアーキテクチャに依存するプログラムを書く必要があるため容易ではない。

そこで本稿では、上記の問題点を改良した移動型プログラム構築環境 (以下、構築環境) の実装方法を提案し、一部は実装を行なった。

2 移動型プログラム

2.1 特徴

通常のプログラムはある特定の計算機で動作する事を前提に作られているため、他種の計算機で動作させるには移植作業が必要になる。また、一度起動したプログラムは起動した計算機上でのみ動作し、他の計算機へ移動することはない。

これに対し移動型プログラムは、複数の計算機間をプログラムごと移動可能なプログラムであり、以下の特徴を持つ。

- プログラムは、計算機間を移動する際、プログラム実行中のスタックやレジスタといった内部状態 (以下、内部状態) を保ったまま移動する。そのため、移動先の計算機上で移動前の処理を再開することが可能である。
- 自律性を持つ。現在の状況を判断しそれに応じた処理が可能である。
- 他の移動型プログラムと対話が可能である。

An implementation of developing tools for mobile program on distributed system.
Naoki Koga*, Yoshiki Ishida**- and Kazuo Ushijima†
*Ungraduate Student, Kyushu University
**Computer Center,Kyushu University
†Graduate School of Information Science and Electrical Engineering,Kyushu University

2.2 利用形態

移動型プログラムの利用方法の１つとして、複数の計算機の情報を利用し何らかの処理を行なう場合を考える。移動型プログラムを使わなければ、ローカルの計算機とそれらの各計算機間で頻繁に通信が発生することになる。しかし移動型プログラムを使うと、プログラムがそれらの計算機を渡り歩きつつ処理をするので、ローカルの計算機と他の計算機間との通信回数が減少する。それだけではなく、一度プログラムを送ってしまえばそれ以外の時間は通信を行なう必要がないため、常時ネットワークに接続していない環境からでもプログラムを実行することが可能である。

3 移動型プログラム構築環境の実装

3.1 実装の方針

今回実装する構築環境には、プラットフォーム非依存である Java 言語を用いる。移動型プログラムを利用する分散環境では、計算機毎に異なる OS やハードウェア構成をしているため、Java 言語を用いることでそれらの違いを吸収し、プログラムの実装や他の計算機への導入を容易にするためである。また計算機のアーキテクチャに依存する手続き (ネイティブメソッド) は利用しない。これは、Java 言語が持っているプラットフォーム非依存性を損なわないようにするためである。

この方針に従い現在構築環境を実装中である。構築環境は「移動型プログラム実行部」と「プリプロセッサ」の２つから成る。

3.2 移動型プログラム実行部

移動型プログラム実行部 (以下、実行部) は、プログラムの移動や各移動型プログラムの実行・管理を行なう。プログラムの移動は実行部同士の通信によって実現する。一方移動型プログラムは、実行部を構成する要素の１つである Mobile クラスのサブクラスという形で利用者が記述し、実行部の上で動作する。

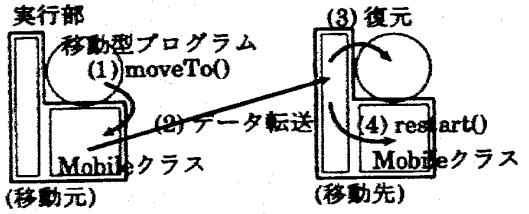

図１ プログラムの移動

プログラムの移動は、Mobile クラスに用意した **moveTo(移動先計算機名, メソッド名)** メソッドを呼ぶことで行なわれる。**moveTo** メソッドは以下のように動作する。

1. 実行部は、現在動作している移動型プログラムのJavaバイトコードを保持している。それを移動先の計算機へ送る。

2. Javaには、現在動作しているオブジェクトをバイト列に変換する、オブジェクト直列化 (Object Serialization) という機能が備わっている。これを用いて移動型プログラムのオブジェクトをバイト列に変換し、それも移動先の計算機へ送る。

3. 移動先の計算機は、受けとった2つの情報を元に移動型プログラムを復元する。

4. 復元した移動型プログラムの **restart** メソッドを呼ぶ (これはMobileクラスに実装する)。この中で、**moveTo** メソッドで指定したメソッドが呼ばれ、移動型プログラムは処理を再開する。

上記1～4の方法で、移動型プログラムのオブジェクトが持つ大局的な変数は保持できる。しかしながら、(A) 移動直前の処理内容や、(B) プログラムがメソッドを呼んだ時に戻り先を保持するスタックは保存できない。そのため、**moveTo** メソッドの処理が終了しても **moveTo** メソッドの呼び出し元へ制御が戻ってこれず、**moveTo** メソッド以降の命令は実行されない。

ただ (A) については、**moveTo** メソッドの前には現在使用中のローカル変数を保存する命令を、移動後実行されるメソッド内の先頭には復元する命令をそれぞれ追加することで状態を保持できる。詳細は次節で述べる。

(B) については、ネイティブメソッドを使わない限りスタック情報を得ることが出来ない。このため今回の方針では保存・復元は不可能である。

3.3 プリプロセッサ

実行部を実装し、Mobileクラスのサブクラスとして移動型プログラムを記述することにより移動型プログラムは動作する。しかしながら、内部状態の保存・復元のコードを移動型プログラム記述者自身で書かねばならないため、プログラムの作成は困難である。また、**moveTo**(移動先計算機名, メソッド名) という記述法では、RPC(Remote Procedure Call) と同じであり、わざわざ移動型プログラムという形で実装する意味がない。

これらの問題を解決するため、内部状態の保存・復元のコードを自動的にソースコードに追加するプリプロセッサを実装する。プリプロセッサの利用手順を図2に示す。プログラムのソースコードに **moveTo**(移動先計算機名) の記述があると、プリプロセッサは自動的に必要な処理をソースコードに追加する。移動先では **moveTo** メソッドの次の文からプログラムの実行を再開する。

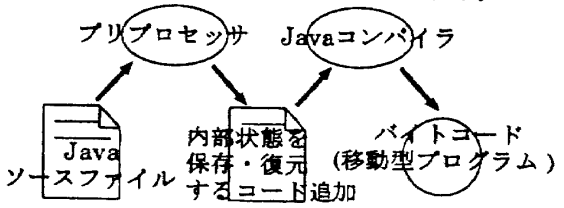

図2 プリプロセッサの利用手順

しかし、ソースコード中の任意の場所に登場する **moveTo** メソッドに対してこのような処理を行なうのは困難なので、いくつかの制限事項を設けた。

- 移動型プログラムを起動するとまず **start** メソッドを実行する。**start** メソッドの処理が終了すると、移動型プログラムの動作が終了する。

- **moveTo** メソッドは **start** メソッドでのみ利用可能。

- **start** メソッド内でも、for, while, if といった制御構文の中では **moveTo** メソッドは利用不可能。

この制限により、メソッドの呼び出し順序を保存する必要がなくなる。以下の方法でプリプロセッサを実装すれば、移動直前のローカル変数の内容も保存・復元が行なえる。

1. 構文解析により **start** メソッドの、名前、引数、返り値の型を記録する。

2. 同様に **start** メソッド全体で有効な変数の名前、型を記録する。

3. **start** メソッド内で **moveTo** メソッドを発見すると、その前後でメソッドを2つに分離するようにソースコードを書き換える。**moveTo** より前の部分には、ローカル変数を保存する記述を加える。**moveTo** より後の部分は新しいメソッドとし、ローカル変数を復元する記述を加える。

プリプロセッサを利用すると、移動プログラム記述者には移動前と移動後で同じメソッドを実行しているように見え、プリプロセッサを使わない方法に比べて記述が容易となる。

4 おわりに

本稿では、移動型プログラムを動作させる実行部と、プログラムのローカル変数の保存・復元を行なうプリプロセッサと2つのプログラムについて、Javaの処理系がもつプラットフォーム非依存性を保った上での実装方法を提案した。また実行部については実装も行なった。

今後はこの構築環境をすべて実装し、その構築環境を用いて作成した移動型プログラムについて様々な評価を行なう必要がある。また、プリプロセッサを設計する段階で課した制限は厳しすぎるため、既存の実装よりも移動型プログラムの記述に対する制限を増やしている。この部分を考慮する必要もある。

移動型プログラムを実際に利用する際には、セキュリティを考慮しなければならない。これは今後実装していく予定である。

参考文献

[1] General Magic : Mobile Agent White Paper
http://www.genmagic.com/agents/Whitepaper/whitepaper.html

[2] IBM Aglets Workbench
http://www.trl.ibm.co.jp/aglets/

[3] Kafka: Yet Another Multi-Agent Library for Java
http://www.fujitsu.co.jp/hypertext/free/kafka/